Проектирование программных систем

HEICA

TEMPUS

Цели

- Содержание этапа проектирования и его место в жизненном цикле конструирования программных систем
- Обзор архитектурных моделей ПО
- Классические проектные характеристики: модульность, информационная закрытость, сложность, связность, сцепление

Особенности процесса синтеза программных систем

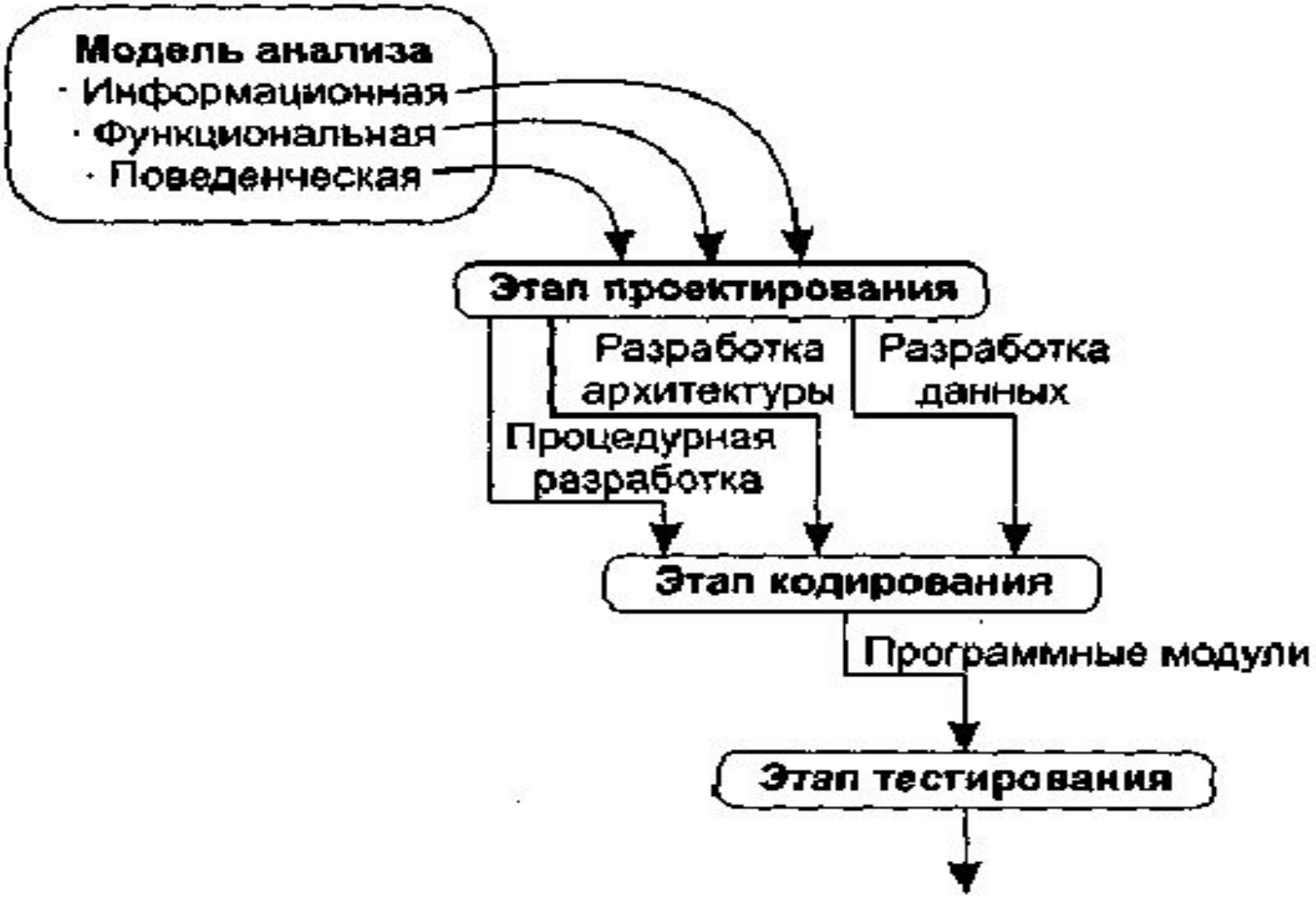
Синтез программных систем

Технологический цикл конструирования ПО включает:

анализ, синтез и сопровождение.

- В ходе анализа ищется ответ на вопрос: «Что должна делать будущая система?». Именно на этой стадии закладывается фундамент успеха всего проекта. Известно множество неудачных реализаций из-за неполноты и неточностей в определении требований к системе.
- В процессе синтеза формируется ответ на вопрос: «Каким образом система будет реализовывать предъявляемые к ней требования?». Выделяют три этапа синтеза: проектирование ПС, кодирование ПС, тестирование ПС

Информационные потоки процесса синтеза ПС

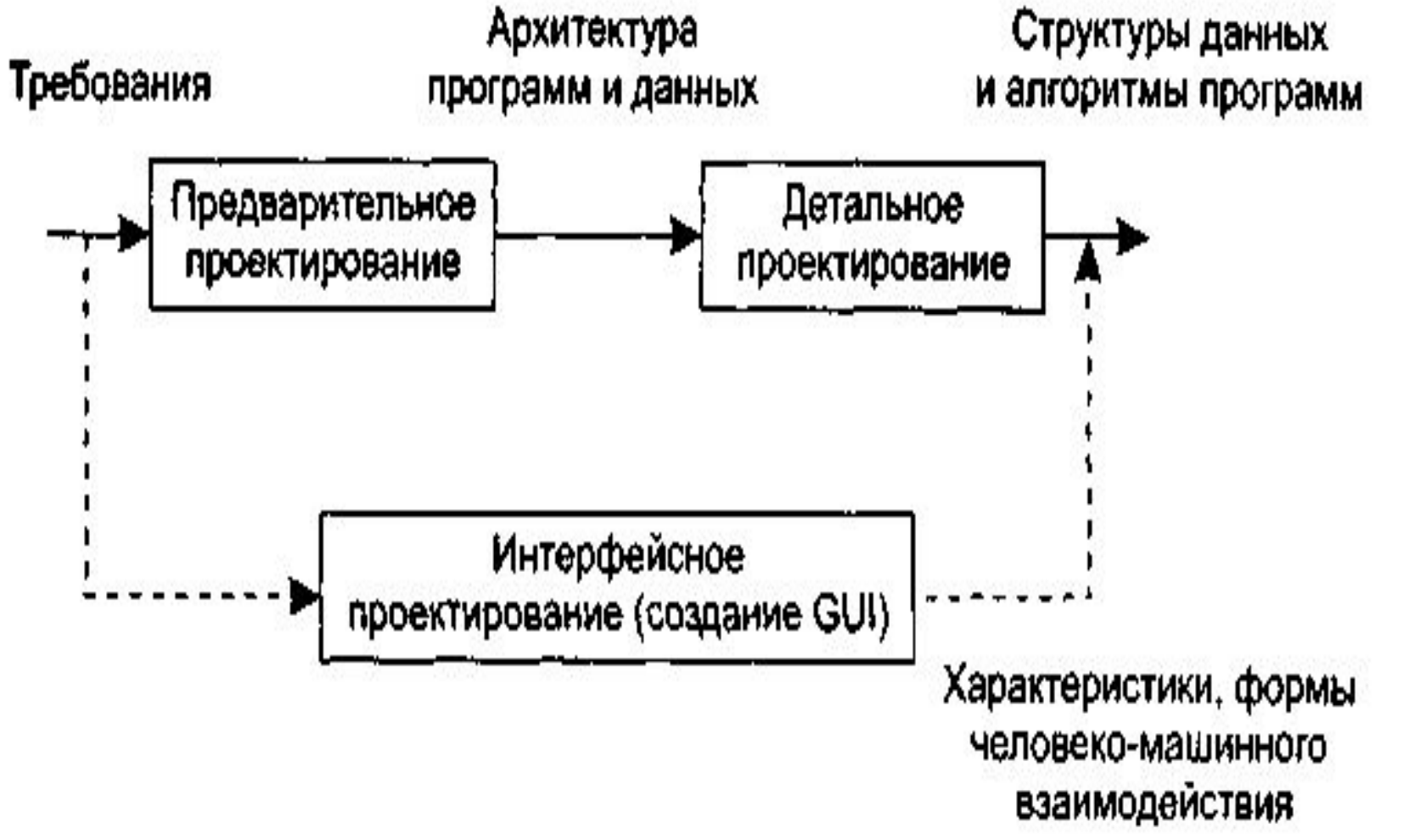


Проектирование

Следует отметить, что решения, принимаемые в ходе проектирования, делают его стержневым этапом процесса синтеза. Важность проектирования можно определить одним словом — качество.

Проектирование — этап, на котором «выращивается» качество разработки ПС. Справедлива следующая аксиома разработки: может быть плохая ПС при хорошем проектировании, но не может быть хорошей ПС при плохом проектировании. Проектирование обеспечивает нас такими представлениями ПС, качество которых можно оценить. Проектирование — единственный путь, обеспечивающий правильную трансляцию требований заказчика в конечный программный продукт

Схема информационных связей процесса проектирования



Предварительное проектирование

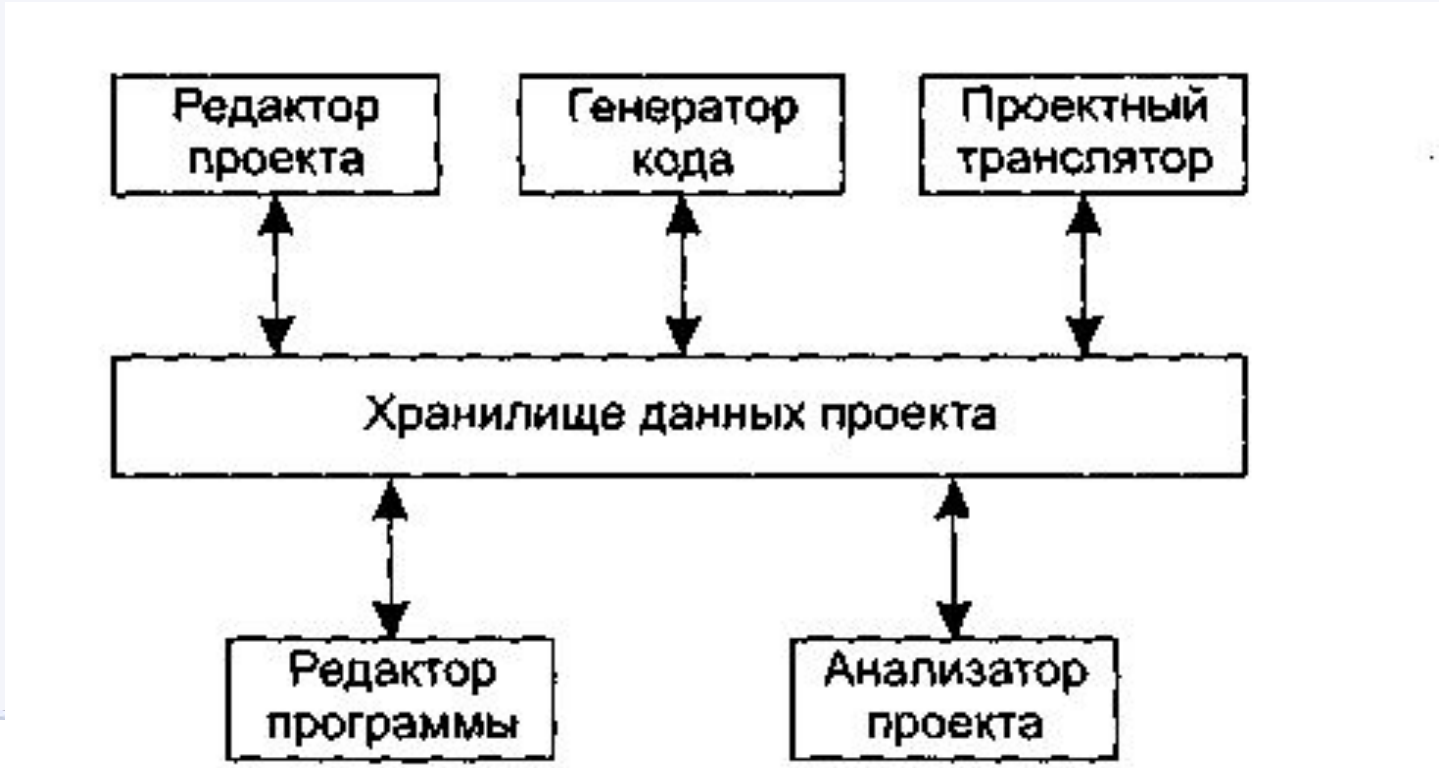
Предварительное проектирование включает три типа деятельности:

1. *Структурирование системы.* Система структурируется на несколько подсистем, где под подсистемой понимается независимый программный компонент. Определяются взаимодействия подсистем.
2. *Моделирование управления.* Определяется модель связей управления между частями системы.
3. *Декомпозиция подсистем на модули.* Каждая подсистема разбивается на модули. Определяются типы модулей и межмодульные соединения.

Рассмотрим вопросы структурирования, моделирования и декомпозиции более подробно.

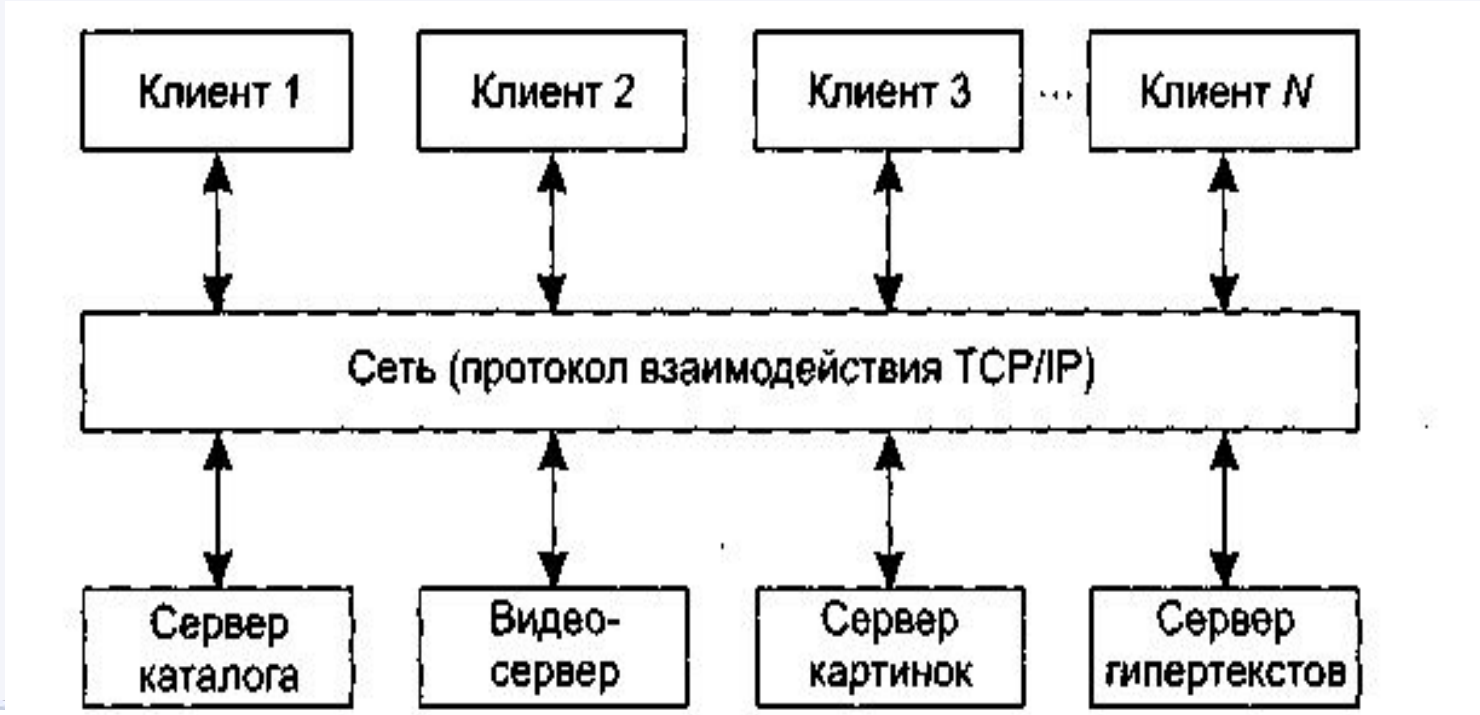
Модель хранилища данных

В *модели хранилища данных* подсистемы разделяют данные, находящиеся в общей памяти. Как правило, данные образуют БД. Предусматривается система управления этой базой.



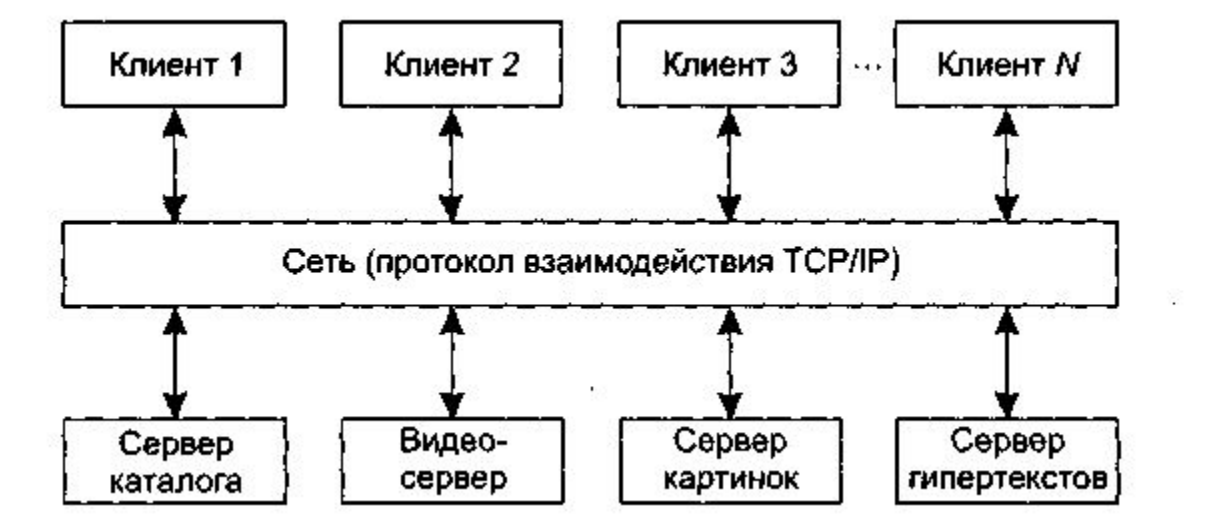
Модель клиент-сервер

Модель клиент-сервер используется для распределенных систем, где данные распределены по серверам. Для передачи данных применяют сетевой протокол, например TCP/IP.



Трехуровневая модель

Трехуровневая модель является развитием модели клиент-сервер.



Трехуровневая модель (2)

Уровень графического интерфейса пользователя запускается на машине клиента. Бизнес-логику образуют модули, осуществляющие функциональные обязанности системы.

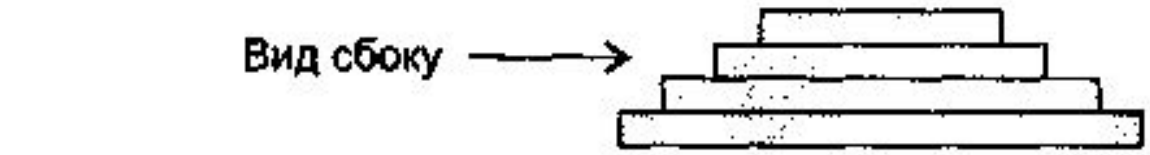
- Этот уровень запускается на сервере приложения.
- Реляционная СУБД хранит данные, требуемые уровню бизнес-логики. Этот уровень запускается на втором сервере — сервере базы данных.

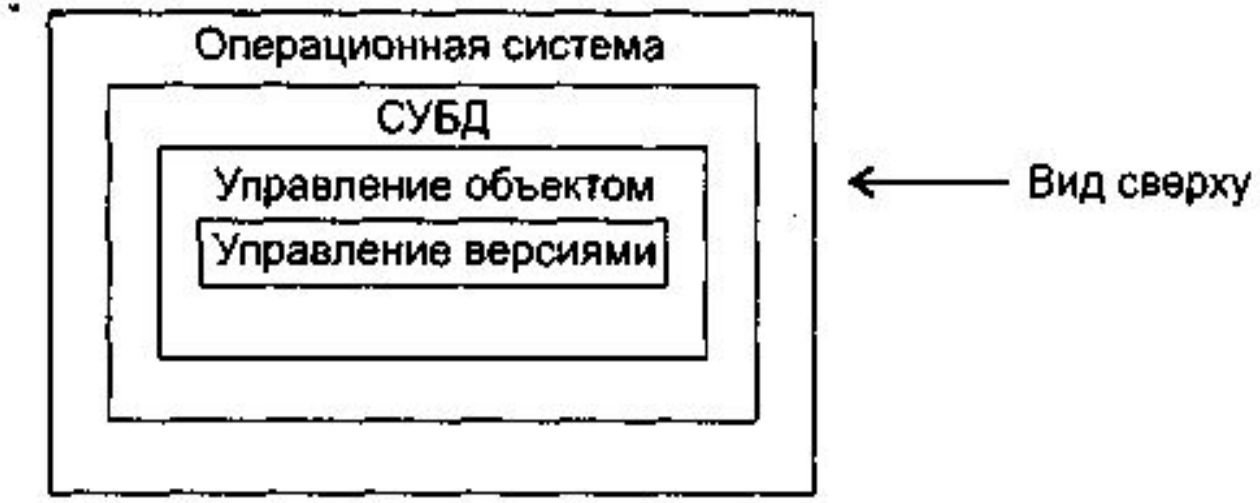
Преимущества трехуровневой модели:

- упрощается такая модификация уровня, которая не влияет на другие уровни;
- отделение прикладных функций от функций управления БД упрощает оптимизацию всей системы.

Модель абстрактной машины

- *Модель абстрактной машины* отображает многослойную систему.
- Каждый текущий слой реализуется с использованием средств, обеспечиваемых слоем-фундаментом.





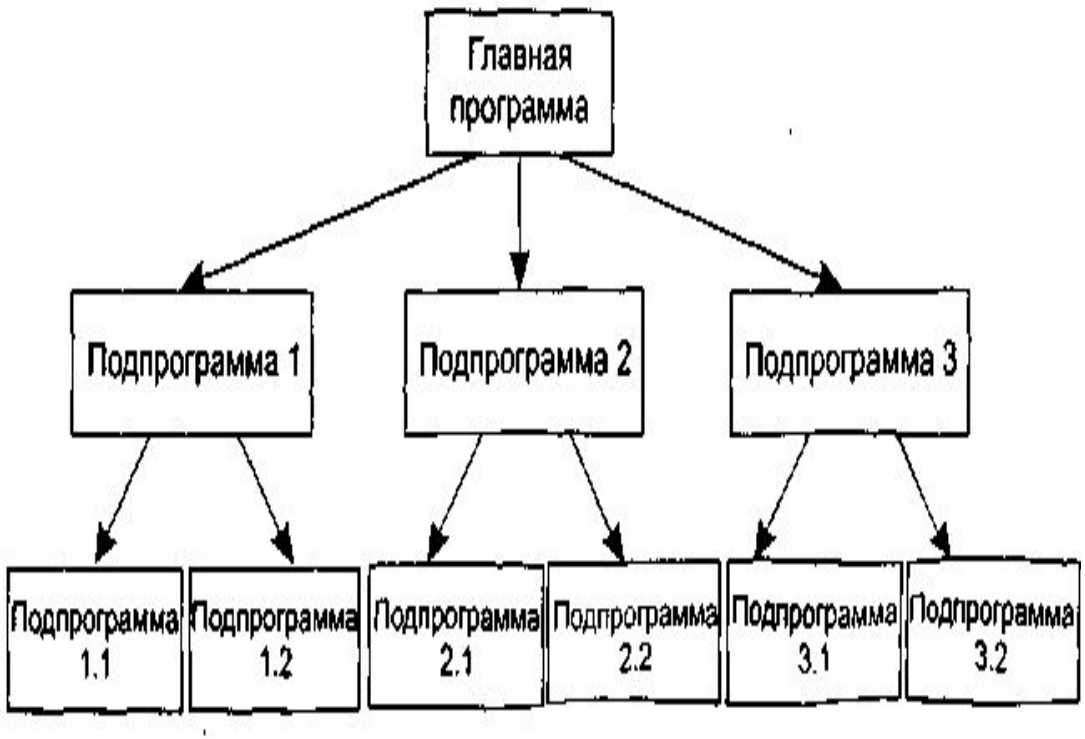
Моделирование управления

Типы моделей управления

- Известны два типа моделей управления:
- модель централизованного управления;
- модель событийного управления.

Модель централизованного управления

В модели централизованного управления одна подсистема выделяется как системный контроллер. Ее обязанности — руководить работой других подсистем. Различают две разновидности моделей централизованного управления: *Модель вызов-возврат* и *Модель менеджера,* которая используется в системах параллельной обработки.



Модель событийного управления

В модели событийного управления системой управляют внешние события. Используются две разновидности модели событийного управления:

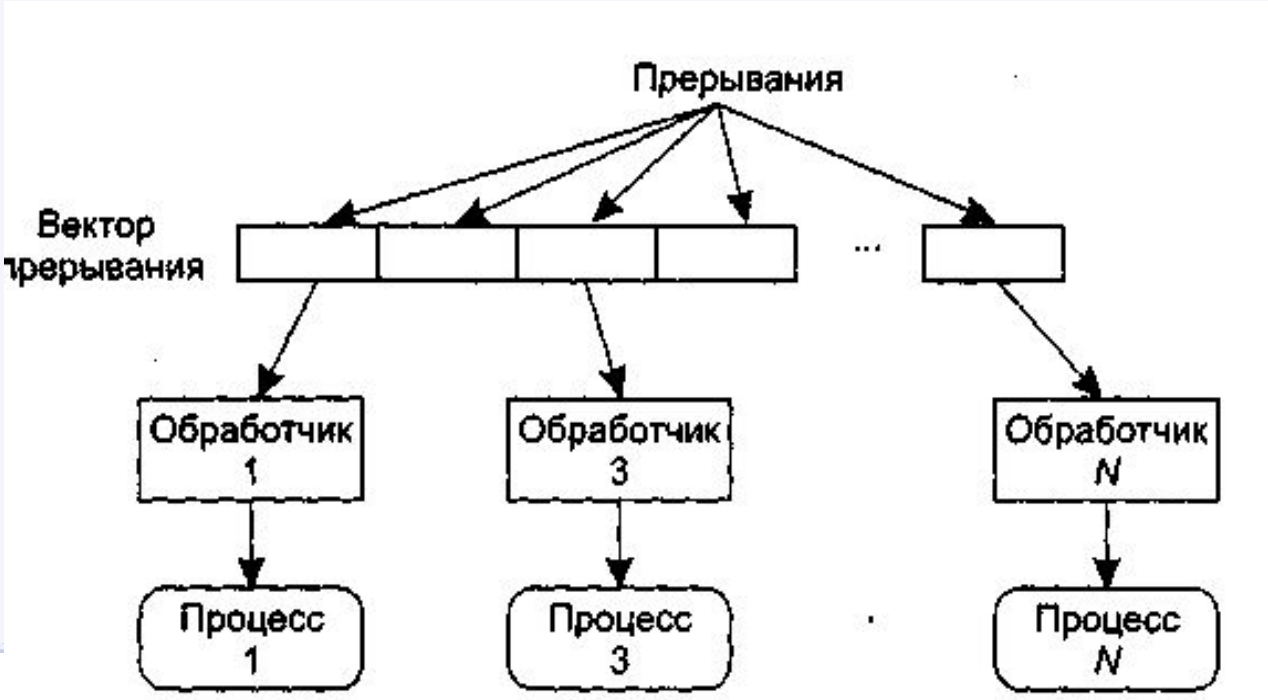
- широковещательная модель:
- модель, управляемая прерываниями.

Широковещательная модель

- В *широковещательной модели* каждая подсистема уведомляет обработчика о своем интересе к конкретным событиям. Когда событие происходит, обработчик пересылает его подсистеме, которая может обработать это событие. Функции управления в обработчик не встраиваются.

Модель, управляемая прерываниями

- *В модели, управляемой прерываниями* все прерывания разбиты на группы — типы, которые образуют вектор прерываний. Для каждого типа прерывания есть свой обработчик. Каждый обработчик реагирует на свой тип прерывания и запускает свой процесс.



Модульность

- Модуль — фрагмент программного текста, являющийся строительным блоком для физической структуры системы. Как правило, модуль состоит из интерфейсной части и части-реализации.
- Модульность — свойство системы, которая может подвергаться декомпозиции на ряд внутренне связанных и слабо зависящих друг от друга модулей.
- По определению Г. Майерса, модульность — свойство ПО, обеспечивающее интеллектуальную возможность создания сколь угодно сложной программы.

Связность модуля (Cohesion)

- Связность модуля (Cohesion) — это мера зависимости его частей . Связность — внутренняя характеристика модуля. Чем выше связность модуля, тем лучше результат проектирования, то есть тем «черней» его ящик (капсула, защитная оболочка модуля), тем меньше «ручек управления» на нем находится и тем проще эти «ручки».

Характеристика связности модуля

Тип связности	Сопровождаемость	Роль модуля
Функциональная	Лучшая сопровождаемость	«Черный ящик»
Информационная (последовательная)		Не совсем «черный ящик»
Кэммуникативная		«Серый ящик»
Процедурная		«Белый» или «просвечивающий ящик»
Временная	Худшая сопровождаемость	«Белый ящик»
Логическая		
По совпадению		

Функциональная связность

Функционально связный модуль содержит элементы, участвующие в выполнении одной и только одной проблемной задачи. Примеры функционально связных модулей:

- Вычислять синус угла;
- Проверять орфографию;
- Читать запись файла;
- Вычислять координаты цели;
- Вычислять зарплату сотрудника;
- Определять место пассажира.

Каждый из этих модулей имеет единичное назначение.

Когда клиент вызывает модуль, выполняется только одна работа, без привлечения внешних обработчиков.

Информационная связность

При информационной (последовательной) связности элементы-обработчики модуля образуют конвейер для обработки данных — результаты одного обработчика используются как исходные данные для следующего обработчика. Приведем пример:

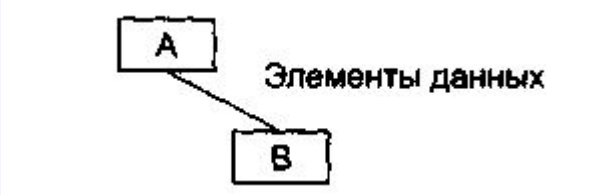
- Модуль Прием и проверка записи
- прочитать запись из файла
- проверить контрольные данные в записи
- удалить контрольные поля в записи
- вернуть обработанную запись
- Конец модуля

В этом модуле 3 элемента. Результаты первого элемента (прочитать запись из файла) используются как входные данные для второго элемента (проверить контрольные данные в записи) и т. д.

Сцепление модулей

- Сцепление (Coupling) — мера взаимозависимости модулей по данным. Сцепление — внешняя характеристика модуля, которую желательно уменьшать.
- Количественно сцепление измеряется степенью сцепления (СЦ). Выделяют 6 типов сцепления.

1.**Сцепление по данным** (СЦ=1). Модуль А вызывает модуль В.

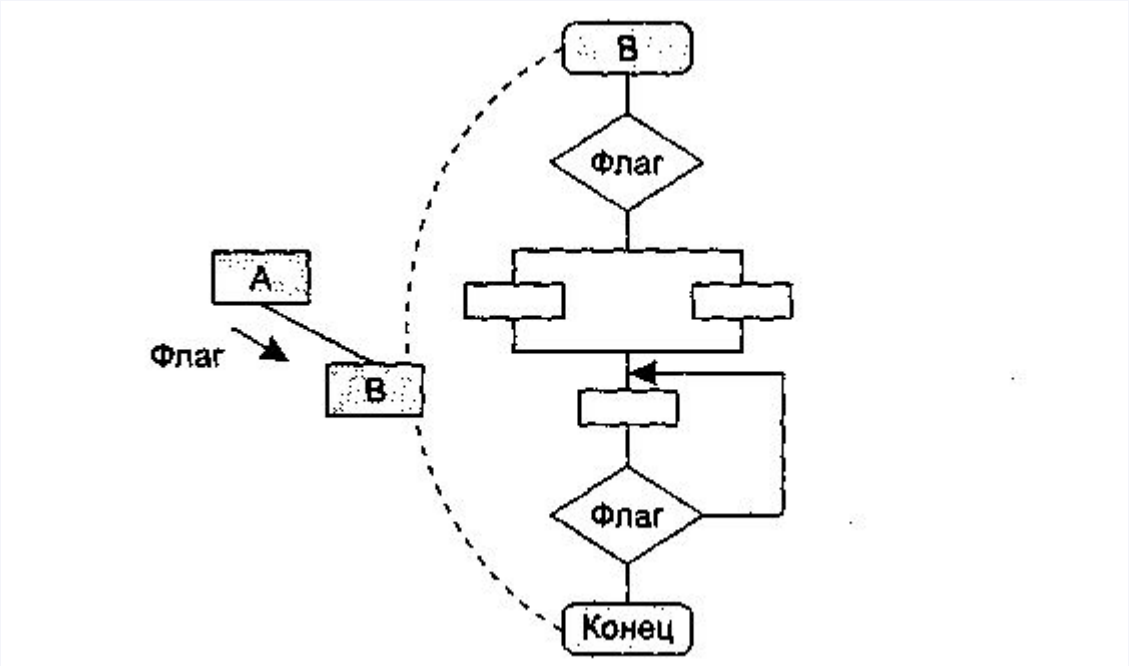


Все входные и выходные параметры вызываемого модуля — простые элементы данных.

2.**Сцепление по образцу** (СЦ=3). В качестве параметров используются структуры данных.

Сцепление модулей (прод.)

3.Сцепление по управлению (СЦ=4). Модуль А явно управляет функционированием модуля В (с помощью флагов или переключателей), посылая ему управляющие данные.



Сцепление модулей (прод.)

4. **Сцепление по внешним ссылкам** (СЦ=5). Модули A и B ссылаются на один и тот же глобальный элемент данных.

5.**Сцепление по общей области** (СЦ=7). Модули разделяют одну и ту же глобальную структуру данных.

6.**Сцепление по содержанию** (СЦ=9). Один модуль прямо ссылается на содержание другого модуля (не через его точку входа). Например, коды их команд перемежаются друг с другом.